Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Authoring Tools for Mesa-3D

Dalai Felinto
Djalma Lucio
Luiz Velho (supervisor)

Technical Report TR-15-07 Relatório Técnico

September - 2015 - Setembro

Authoring Tools for Mesa-3D

Dalai Felinto, Djalma Lucio, Luiz Velho

August 31, 2015

Abstract

Mesa-3D is a device for augmented reality experienced in a support surface, traditionally a table. Mesa-3D is part of the PlanoVision system, which was originally intended for real footage from a static point of view experience. This system expands the experience from PlanoVision by allowing the user freedom of movement. Therefore the authoring and deployment tools for this platform must attend the new requirements, such as head tracking and real-time graphics. Blender 3D was used as a case-study on the design and evaluation of such platform.

1 Overview

Mesa-3D bridges virtual reality with a physical environment. The setup includes a state of the art head tracking system, 3D stereo glasses, a 3D digital projector, and a game engine. This allows for dynamic content to be projected on-the-fly and experienced by the user as if they were "real". The illusion built by the Mesa-3D is based on projecting virtual objects on the real table, rendered with the correct perspective from the user point of view. In order to sustain this illusion we should be able to provide content that can be explored fully at any angle. While the industry is moving towards 360° real footage capture and display, there is still no open framework for this. On the other hand, game engines have been around for many years, and are entirely capable of generating correctly projected content on demand. At the same time, 3D hardware is more and more accessible, allowing the editing station to preview the original content even before we deploy it to the Mesa-3D.

Our choice of authoring software fell on Blender 3D [1]. We based our decision on the flexibility that Open Source brings to the table, as well as our expertise and familiarity with the program. Blender 3D and the Blender Game Engine were used extensively to validate the deployment pipeline and serve as a base for further implementations in other engines. That said, Blender 3D can always play a role on the authoring process given its 3D previewing features and importing and exporting capabilities.

1.1 Mesa-3D

Mesa-3D is a technology developed by Visgraf/IMPA. The system core includes a 3D digital projector, 3D stereo glasses, and a head-tracking system. On top of that, external devices, such as a WiiMote controller, can be used to enhance the interaction with the table.

Mesa-3D is an extension of the PlanoVision system [2], which was originally intended for steady users experiencing a pre-captured footage or ray-traced rendered image. This system expands the experience from PlanoVision by allowing the user complete freedom of movement. Both technologies may converge in the future, once 360° capture becomes more accessible and standardized.

1.2 Blender 3D

Blender 3D is a 3D open source platform known for its animation, and rendering capabilities. A less famous component of Blender 3D is its built-in interactive game engine. The Blender Game Engine [3] supports modern GLSL shaders, skinning animation and Python scripting among other capabilities. Despite the name, the Blender Game Engine excel at interactive presentations, scientific visualization and architecture walkthroughs.

Blender 3D was used for editing the 3D content, as well as displaying them interactively using the Blender Game Engine. Since its version 2.75 Blender 3D supports a complete pipeline for stereo 3D movie-making [4].

2 Implementation

We built a small template file, containing:

- Virtual model of the Mesa-3D, for placement of the virtual objects
- Camera at the origin of the file, to be controlled via head tracking
- Python script that is called from the only required logic brick of the scene

2.1 Head Tracking

We use a dedicated OptiTrack server running Motive. The tracked sensors are attached to a cap, easily wearable by the user. For the Mesa-3D system the orientation of the tracked head is not relevant, but only its position. Since the tracking is performance critical, the head position is sent via VPRN to the graphic station running the Mesa-3D software.

2.2 Projection Matrix

For the virtual elements to appear in a correct perspective, we need a special projection matrix. The matrix is calculated considering that the user is facing towards the center of the projection surface at all times. In order to calculate the rendering projection matrix we need:

Head Position The head position relative to the center of the projection

Interocular Distance The separation of the stereo eyes

The head position **P** is constantly updated from the head tracking system. The **width** and the **height** of the table are intrinsic parameters of the system. The interocular distance **I** can be tweaked to better fit the user's eyes. The near **n** and far **f** parameters are intrinsic of the virtual camera, and adjusted accordingly to the desired camera clipping.

We start by using the head position to determine the vector **V** connecting both eyes:

$$\vec{xy} = (P_x, P_y, 0)$$
$$\vec{V} = |\vec{xy} \times (0, 0, 1)|$$

In fact we will need not one but two projection matrices. The left one uses as position: $\vec{eye} = \vec{xy} + |\vec{V}| \cdot \frac{I}{2}$, while the right one: $\vec{eye} = \vec{xy} - |\vec{V}| \cdot \frac{I}{2}$. We then calculate the frustum planes with the **scale** as $\frac{n}{P_z}$:

$$l = scale \cdot (\frac{-width}{2} - \vec{eye}_x)$$
$$r = scale \cdot (\frac{width}{2} - \vec{eye}_x)$$
$$t = scale \cdot (\frac{height}{2} - \vec{eye}_y)$$
$$b = scale \cdot (\frac{-height}{2} - \vec{eye}_y)$$

The projection matrix is then built in its canonical form:

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

2.3 ModelView Matrix

Beside the projection matrix, we also need a unique modelview matrix for Mesa-3D. In order to build The modelview matrix we need:

World Translation The position of the camera in the virtual world, M_{camera}

Eye Position The position of the eye in the real world

We calculate the eye matrix, M_{eye}, using the eye position, $\vec{eye}$, as calculated for the projection matrix, and the vertical component of the Head Position, P_z:

$$M_{eye} = \begin{bmatrix} 1 & 0 & 0 & \vec{eye}_x \\ 0 & 1 & 0 & \vec{eye}_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = M_{camera} \cdot M_{eye}$$

Since the eye position, $\vec{eye}$ is different for each eye, we will have two modelview matrix, just like the projection matrices.

2.3.1 Scaling Control

Sometimes the model to be explored in the Mesa-3D cannot be displayed in its real size. In those cases we need to scale the virtual scene. In other cases we want to dynamically change the scale of the virtual scene to alternate between observing the details and apreending the overall context. From the implementation perspective, the scale affects the modelview matrices, as well as the speed movement in the virtual scene. For those the modelview matrix formula is then:

$$M = M_{camera} \cdot M_{eye} \cdot scale$$

3 Conclusion

The proposed integration into the target software was a success. The non-intrusive approach is flexible enough to be ported to many game engines. As long as the rendering pipeline allows for custom matrices, and there is vrpn support (or similar connectivity with a head-tracking system).

References

[1] http://www.blender.org, visited on August 27th, 2015.

[2] B. Madeira, P. Rosa, C. Volotao, and L. Velho, *Horizontal stereoscopic display based on homologous points*, in Proceedings of VISAPP, 2015.

[3] D. Felinto and M. Pan, *Game Development with Blender*, CENGAGE Learning, 2013 - 434 pages.

[4] https://www.blender.org/manual/render/workflows/multiview.html, visited on August 28th, 2015.